COMP 282

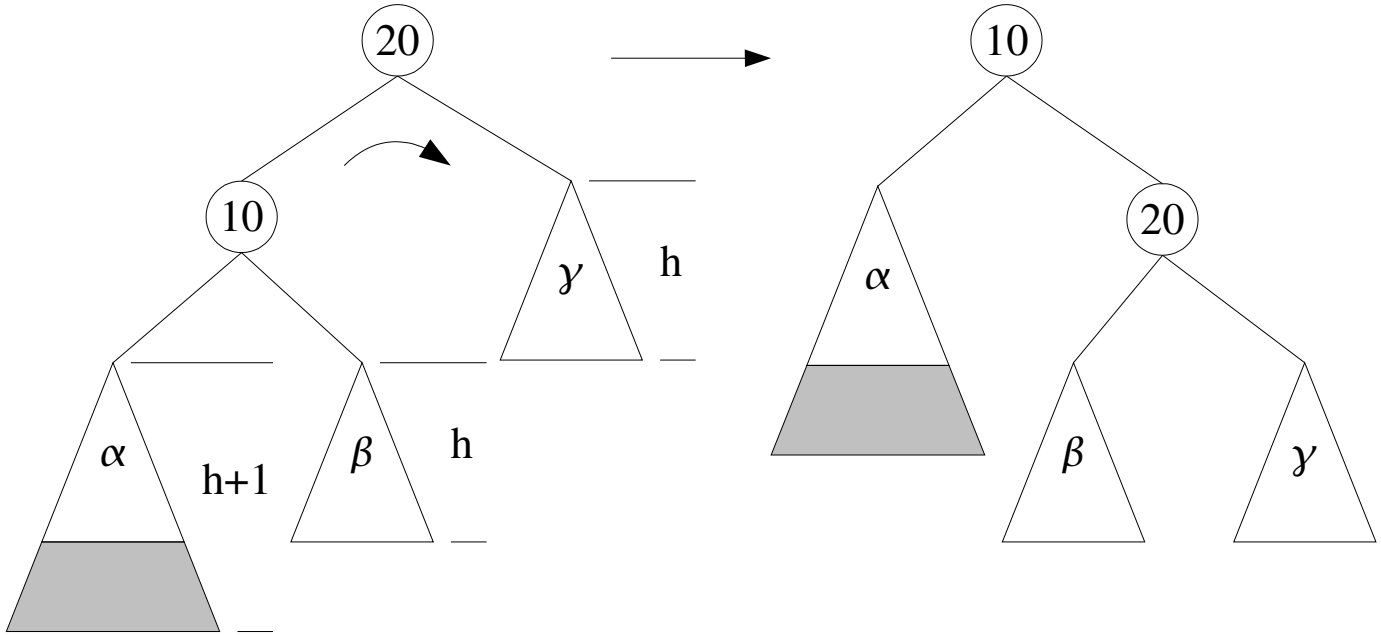
AVL Details
Basic Implementations and Insertion

AVL Node class

```
Class AVLNode
{
   private Object stored;
   private Comparable key;

   private AVLNode right_subtree;
   private AVLNode left_subtree;

   private int height;

   public AVLNode(Object stored_arg, Comparable key_arg);

   private boolean isBalanced();
   private int identifyBalancingAction();
   private rotateRight();
   private rotateLeft();

   public void insert(Object stored_arg, Comparable key_arg);
   public Object search(Comparable key_arg);
   public Object delete(Comparable key_arg);
}
```

Problems requiring single rotation

- When the addition causes the outside subtree to become too tall...



A single rotation will correct the problem.

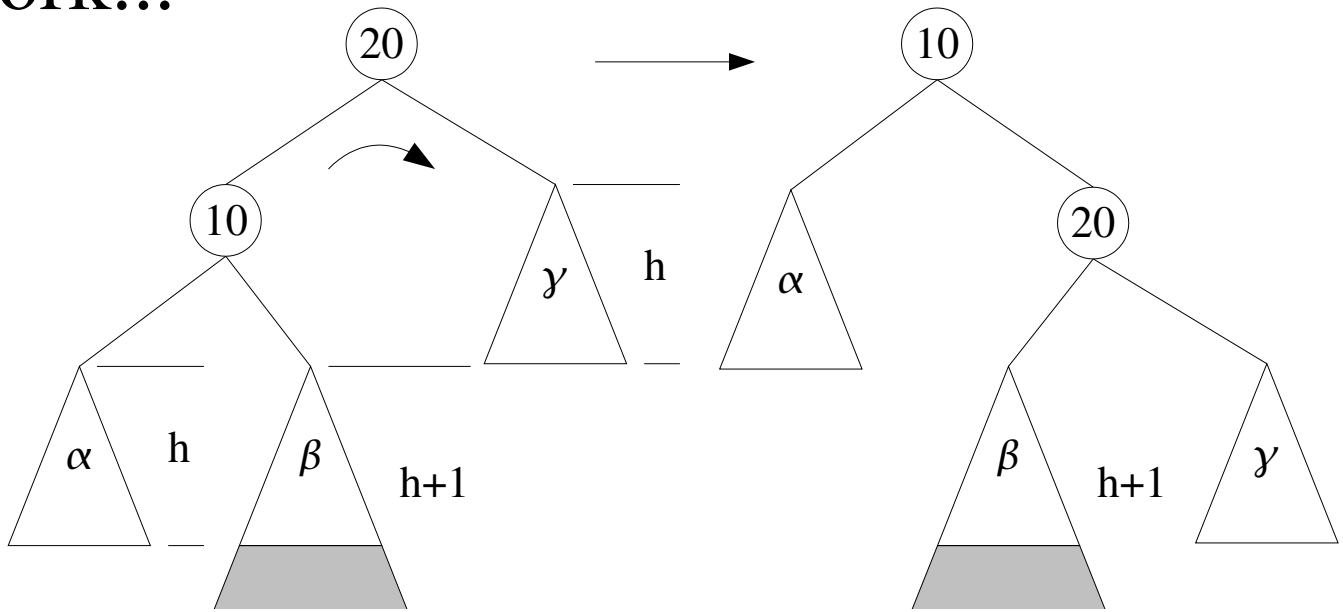
Items of interest

- There are two such cases:
  - The case shown, where “outer” is defined as the left subtree of the left subtree. fixed by a single right rotation at the root node of the unbalanced subtree.
  - The case not shown, where “outer” is defined as the right subtree of the right subtree. (mirror of the case shown) Fixed by a single left rotation.
- This is the most basic case that is fixed by a single rotation.

- After the rotation the height of the resulting tree remains the same as before the insertion.
  - This means that once this imbalance has been identified and corrected no further imbalances will occur.

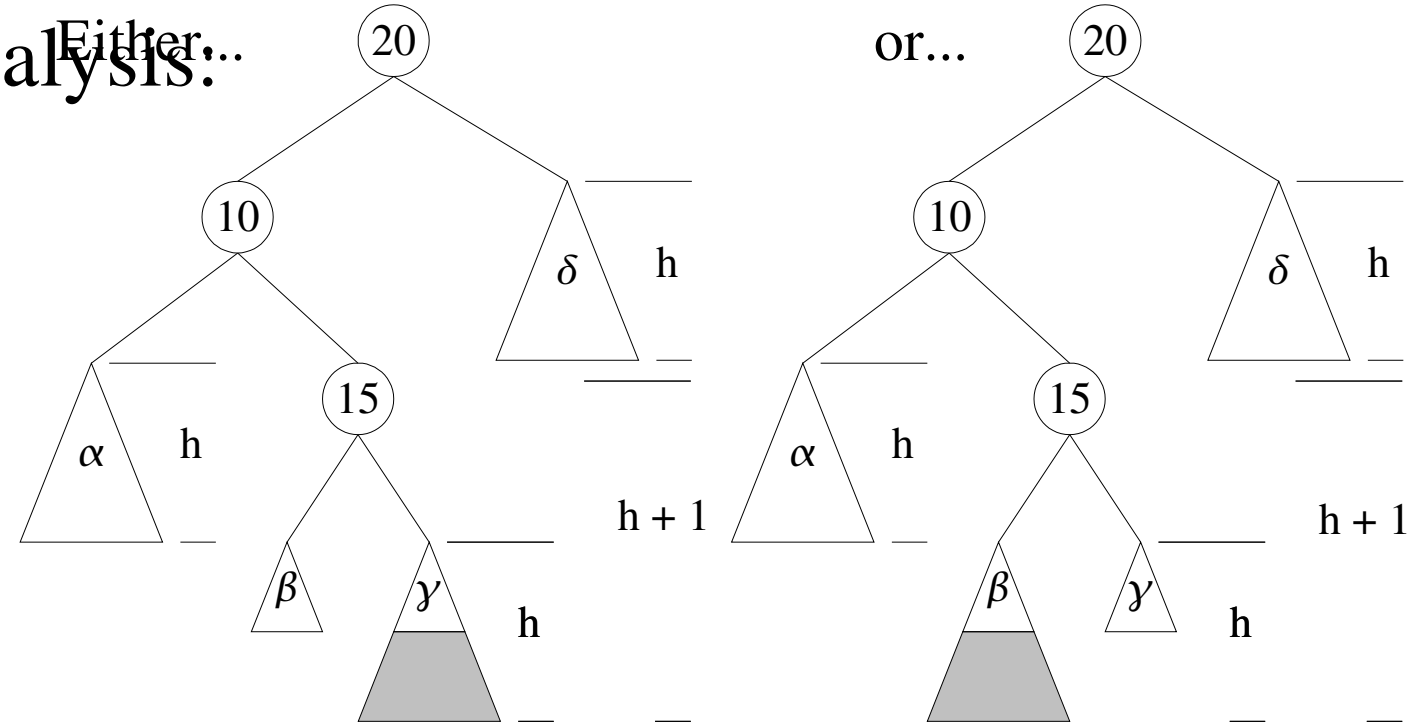
What about “inner” trees?

- If the height difference is dominated by the “inner” subtree then a single rotation doesn't work...



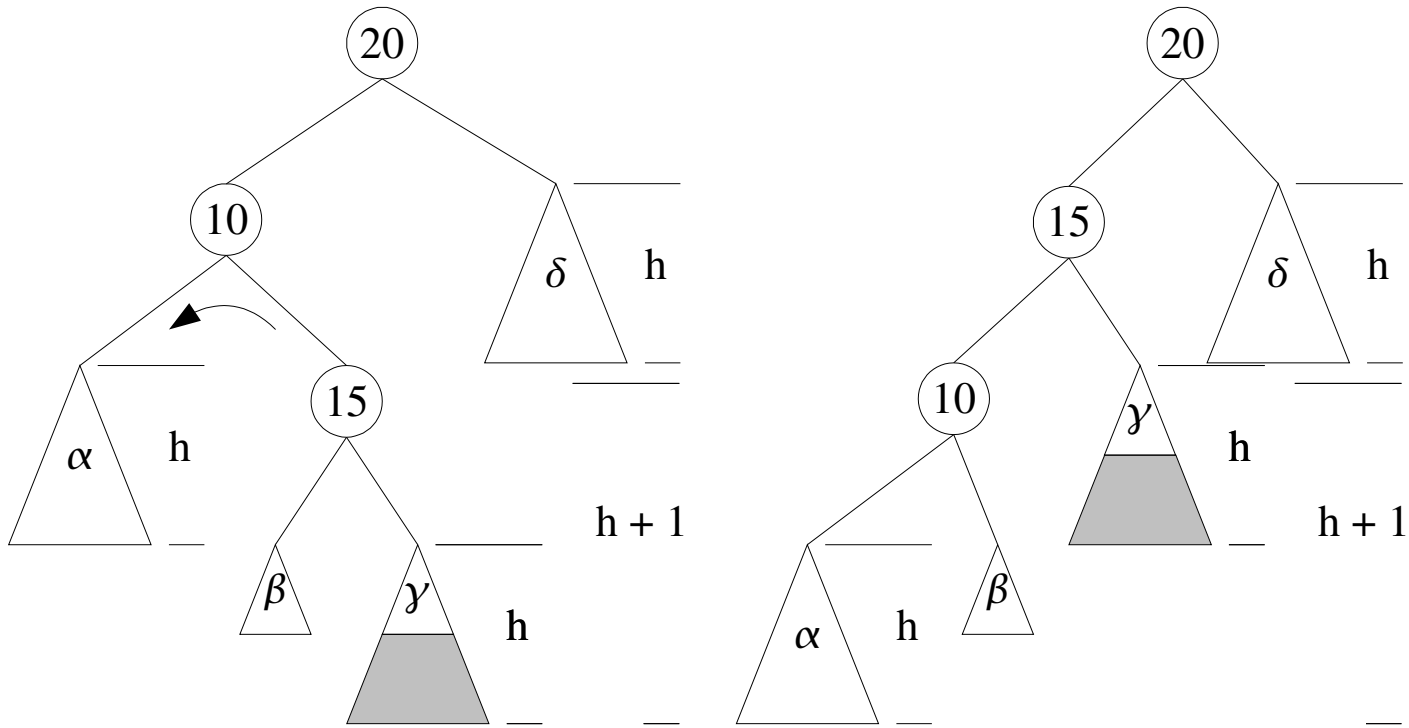
Take a more detailed look...

- The solution is found in a more detailed analysis:



The first scenario

- Transfer the “height” to the outside.

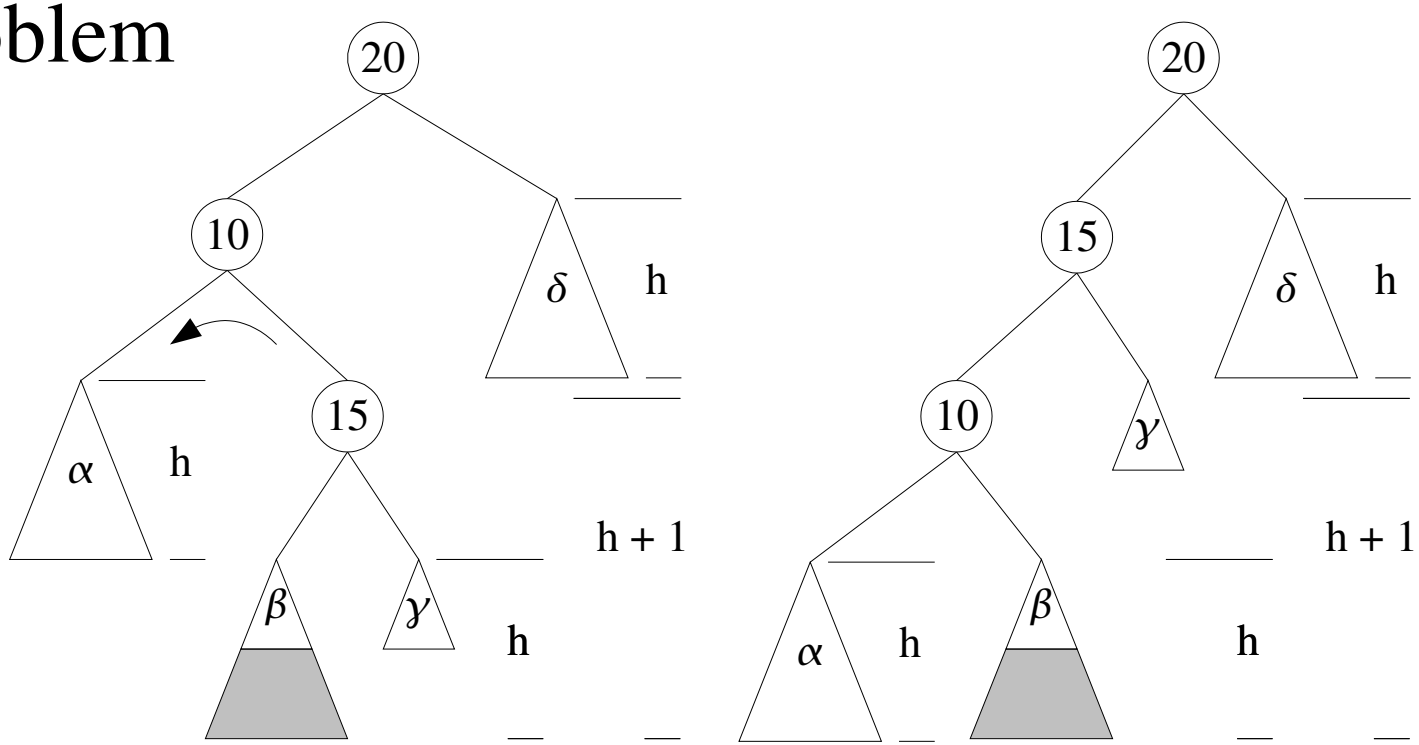


This is good...

- A left rotation performed on the subtree first is sufficient to transform the problem into one that is solved by a single right rotation about the root.
  - (“root” of the unbalanced subtree)

The Second case...

- Trying the same left rotation produces a problem

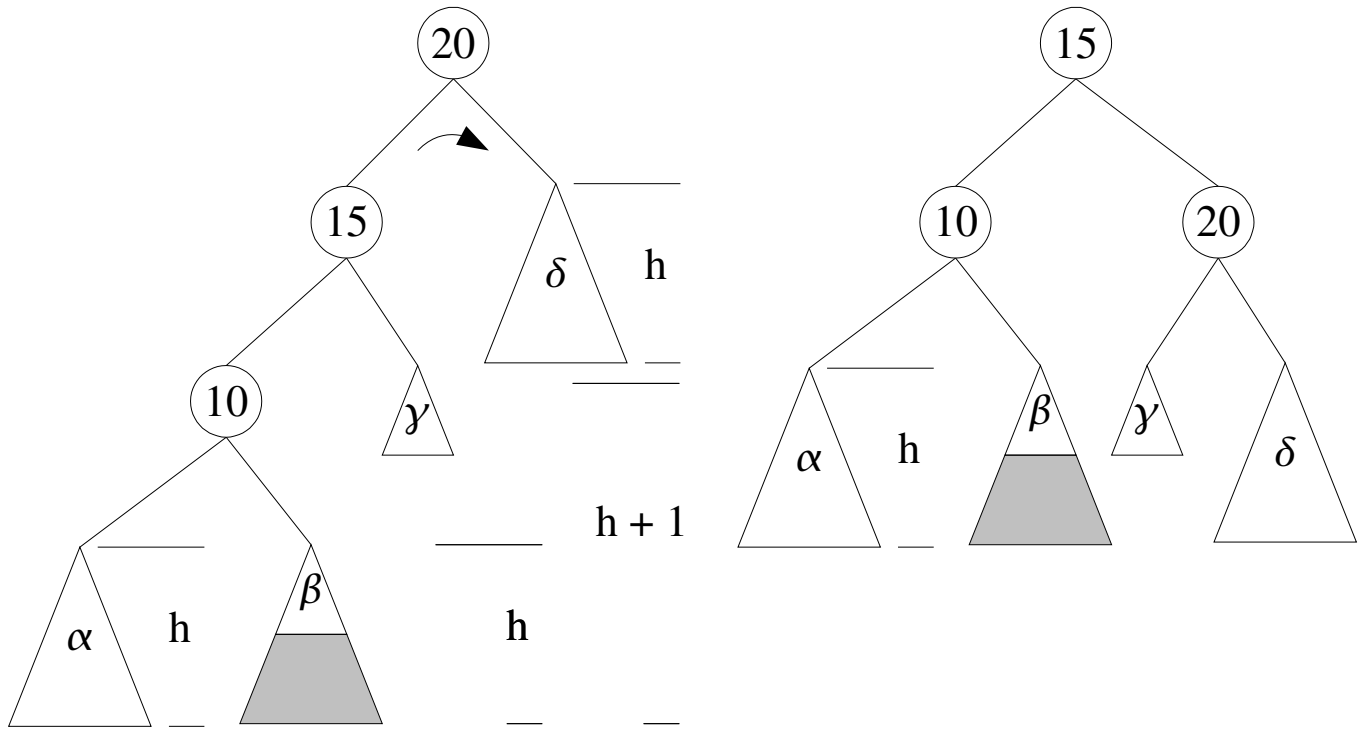


The (temporary) problem

- By performing the left rotation the left subtree becomes unbalanced.
- This problem is only temporary though...
- By performing the same right rotation at the root will correct the unbalanced subtree and the unbalance that existed at the root to begin with...

Kill two problems with one stone

- The same second rotation still works



Two Rotations fix the problem

- Regardless of which case you started with where the “inner” subtree was the problem
- A left rotation at the subtree followed by the right rotation at the root corrects the problem.

Mirror cases

- There are actually four cases to consider that require two rotations to fix:
- The two cases involving height dominated by the left subtree as have been illustrated
  - These are corrected by a left rotation followed by a right rotation.
- Two more cases that are the mirror images of the two presented (height dominated by right subtree)
  - Corrected by a right rotation followed by a left rotation.

How much work does this require?

- In all three cases (and the additional three mirror cases) The height of the tree after the rotations is the same as the height of the tree before the insertion was performed.
- The work then is limited to:
  - performing a single rotation at one node in the tree, or
  - performing two rotations. The first at a node in the tree and the second at the parent node that is the target of the first rotation.